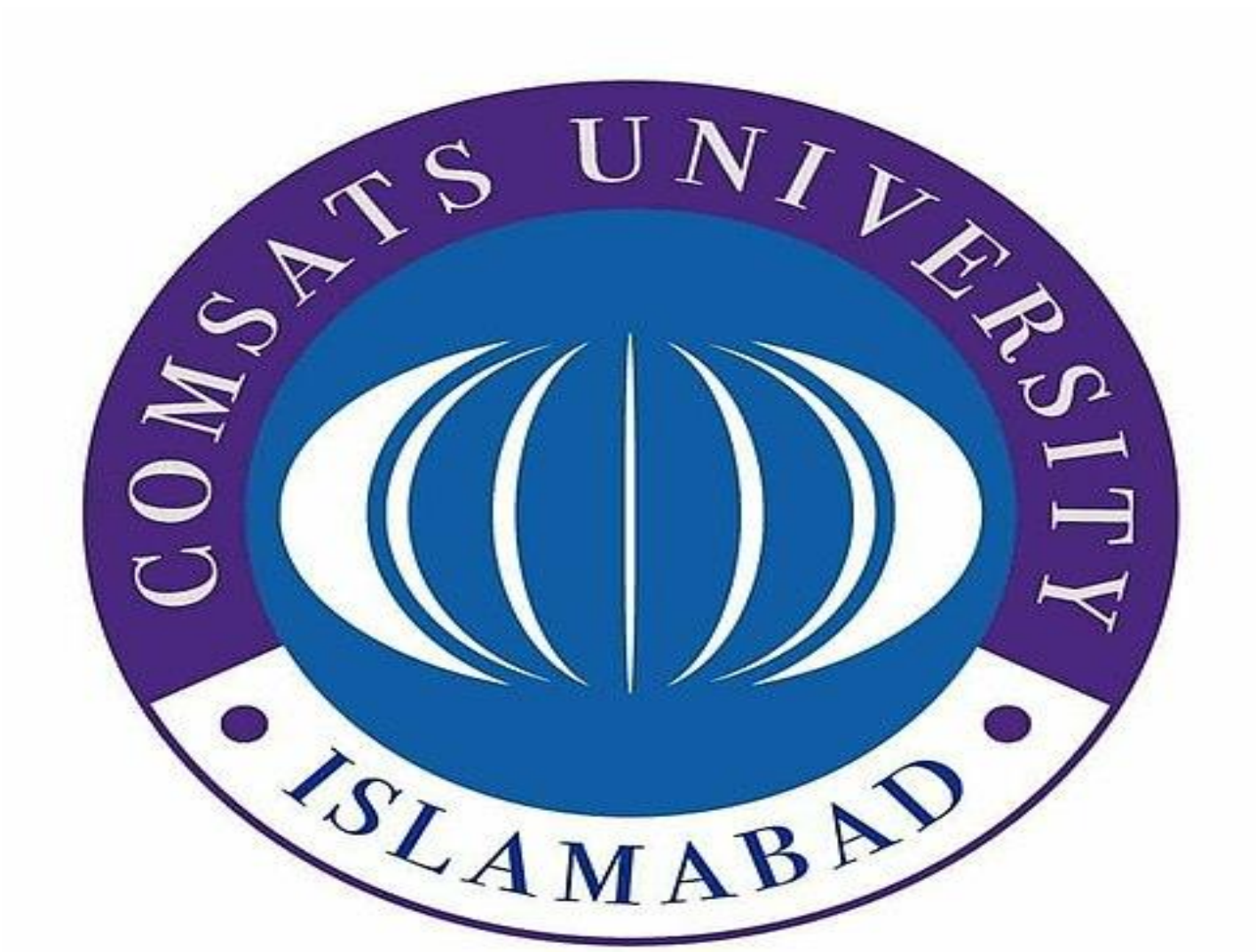


BACHELOR OF SCIENCE IN MATHEMATICS

BY

ARSHAD ISLAM

IQRA BAIG AND

DEPARTMENT OF MATHEMATICS

COMSATS Institute of Information Technology Islamabad – Pakistan

JUN 2022

COMSATS Institue of Information Technology Islamabad

A project presented to

COMSATS Institute of Information Technology, Islamabad in partial fulfillment Of the requirement for the degree of

BACHELOR OF SCIENCE IN MATHEMATICS

By

ARSHAD ISLAM

SP20-BSM-010

IQRA BAIG

SP20-BSM-019

NUMERICAL COMPUTATION: METHODS FOR SOLVING EQUATIONS

An undergraduate project submitted to the

Department of Mathematic

As a partial fulfillment for the semester

BACHELOR OF SCIENCE IN MATHEMATICS

Name	Registration no
Iqra Baig	Sp20-bsm-019
Arshad Islam	Sp20-bsm-010

Professor:

Prof. Dr. Umar Umair

Department of Mathematics CUI, Islamabad

CUI, Islamabad

Signature:

Arshad Islam Registration No.: CIIT/sp20-BSM-010/ISB

Iqra baig Registration No.: CIIT/sp20-BSM-019/ISB

Signature:

Name: Arshad Islam

Signature:

Name: iqra baig

COMSATS UNIVERSITY OF ISLAMABAD

Islamabad - Pakistan

DEDICATION

I dedicate this project to my family, especially to my mother, who pray for me every time and have always been a source of strength for me. I also dedicate this work to my teachers without whose cooperation this work may not be completed by me.

ARSHAD ISLAM

IQRA BAIG

ACKNOWLEDGEMENT

First of all we owe all the appreciation to All Mighty Allah the most beneficent and the most merciful, the creator of this universe, from the vast extent of the outer space to the tiny speck of sand and everything within including you, we and all the brains to think over the most complicated matters of this universe. And then to the one for which everything in this world was created, who was sent here as a mercy to all mankind, the one dearest to Allah, Prophet MUHAMMAD (Peace Be Upon Him).

Moreover, we are under a debt of gratitude to our **Prof.Dr.UmarUmair,** for his teaching greatly redounds to this project. He has always been the most cordial and cooperative to me throughout the research work and the compilation of this desertion. It has been a privilege and great honor for me to work under his supervision.

Especially, the credit goes to our Father and Mother who provided us all the facilities. It was indeed not possible without their prayers.

In the end, we thank all those people who supported us throughout our work specially our teachers, our friends who were, are and will always be my strength.

Contents

1. Introduction

Numerical analysis is concerned with all aspects of numerical solution of a problem. When we are presented with problem that can't be solved directly, then we try to replace it with nearby problem that can be solved more easily. For instance use in developing numerical integration method and root finding method.

The Fundamental concern of numerical analysis is error, its size and its analytical form. When approximating a problem it is prudent to understand nature of error in computed solution. For instance, consider a problem

$P(X)= x^7-25x^6+37x^5+1320x^4-332x^3+42x^2+563+45$

Such problems are unstable or ill condition with respect to ill conditioned with respect to root finding problem.

So, to overcome this problem there is numerical analysis that is the process of obtaining a solution is to reduce original problem to a repetition of the same step so that computation become automatic. Such process is called numerical method. Numerical method which can be used to solve problem called algorithm. It considers all the errors that may affect the result. It consider how much accuracy is required, estimate magnitude of round off. Determine how much iteration is required.

2 Methods to solve Non- linear equations.

A system of nonlinear equations is a system of two or more equations in two or more variables containing at least one equation that is not linear. Recall that a linear equation can take the form Ax+By+C=0. Any equation that cannot be written in this form in nonlinear. The substitution method we used for linear systems is the same method we will use for nonlinear systems. We solve one equation for one variable and then substitute the result into the second equation to solve for another variable, and so on. There is, however, a variation in the possible outcomes.

So there are some methods in numerical analysis

1. ***Bisection Method***
2. ***Regular false method***
3. ***Newton Raphson method***
4. ***Secant method***
5. ***Fixed point method***

2.1 BISECTION METHOD

This method based on repeated application of intermediate value property

An equation f(x) =0 where f(x) is real continuous function.
It has at least one root between a and b if f (a) f (b) <0

In Numerical analysis (methods), Bisection method is one of the simplest, convergence guaranteed method to find real root of non-linear equations.

Bisection method also known as Bolzano or Half Interval or Binary Search method has following merits or benefits:

Merits

Convergence is guaranteed:

Bisection method is bracketing method and it is always convergent.

Error can be controlled:

In Bisection method, increasing number of iteration always yields more accurate root.

Does not involve complex calculation

Bisection method does not require any complex calculations. To perform Bisection method, all we need is to calculate average of two numbers.

Guaranteed error bound:

In this method, there is a guaranteed error bound, and it decreases with each successive iteration. The error bound decreases by ½ with each iteration.

Bisection method is very simple and easy to program in computer.

Bisection method is fast in case of multiple roots.

Demerits

- Biggest dis-advantage is the slow convergence rate. Typically bisection is used to get an initial estimate for such faster methods such as Newton-Raphson that requires an initial estimate. There is also the inability to detect multiple root.
- Although the Bisection method's convergence is guaranteed, it is often slow.
- Choosing a guess that is close to the root may necessitate numerous iterations to converge.
- Some equations' roots cannot be found. Because there are no bracketing values, like $f(x) = x^2$.
- Its rate of convergence is linear.
- It is incapable of determining complex roots.
- If the guess interval contains discontinuities, it cannot be used.
- It cannot be applied over an interval where the function returns values of the same sign.

ALGORITHM

- Select interval [a,b] with root inside
- Split root into two where Xm=($\frac{a+b}{2}$)
- If f(a) and f(Xm) have different values the b=Xm Otherwise a=Xm in next iteration .
- Find new estimate of the root Xm, then find absolute relative approximate error
- |error|=|$\frac{X_m^{new}-x_m^{old}}{x_m^{new}}$|*100
- Repeated from step 2 until |approximate error |< expected error
- Root=Xm

Code

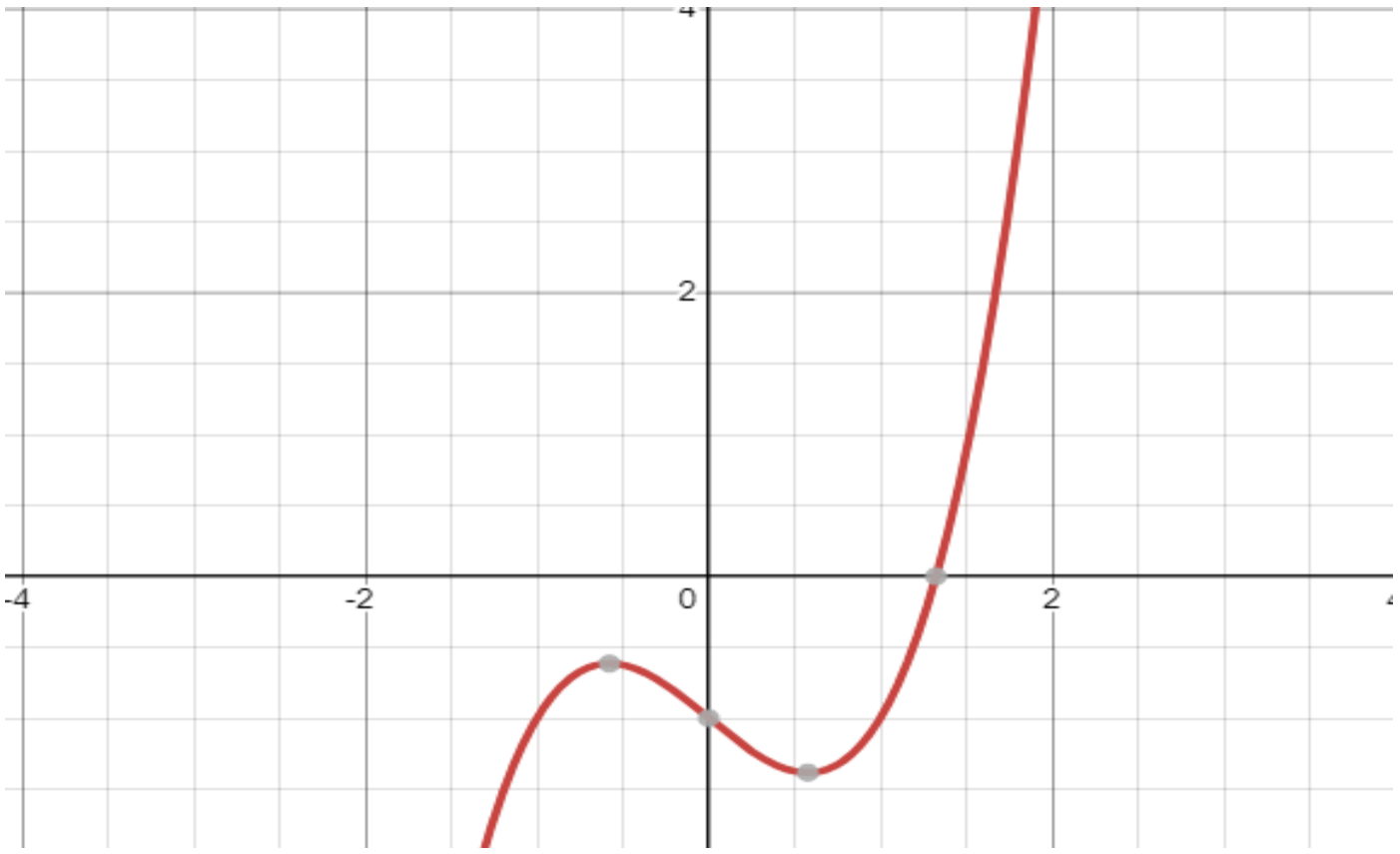
```
a#guess1=1,guess2=2
from math import sin
def bisection(x0,x1,e):
step=1
condition=True
while condition:
x2= (x0+x1)/2
print('iteration %d, x2 = %0.6f and f(x2)= %0.6f' %(step,x2,f(x2))) #
if f(x0) * f(x2) < 0

x1=x2
else:
        x0 = x2 #
step=step+1
condition=abs(f(x2))>e
print('rootis:%0.8f'%x2)
returnx2

deff(x):
returnx**3-x-1
x0=float(input('firstguess:))
x1=float(input('secondguess:'))
e=float(input('tolerance:'))
if f(x0) * f(x1) > 0.0:

print('given guess values do not bracket the root')

else:
```

root = bisection(x0,x1,e) # if above condition is not true compiler will go to this step at above

Consider a problem

f(x)=x^3-x-1

Graph



f(x)=x^3-x-1

Iteration 1

Step 1:

Take guesses

f(1)=1-1-1=-1

f(2)=8-2-1=5

So we have a and b are 1, 27

Step 2:

$x_m = \frac{(a+b)}{2}$

$x_m = \frac{(1+2)}{2}$

Xm=1.5

Step 3

$f(1.5) = (1.5)^3 - 1.5 - 1$ ==7/8

At 1.5 function gives 7/8 value now root exist between 1 and 1.5

2nd iteration

Xm=($\frac{(1+1.5)}{2}$)=1.25

$f(1.25)$)=$\frac{-19}{64}$

Now root exist between 1.25 and 1.5

Error approximation

$|\epsilon| = |\frac{1.25-1.5}{1.25}| * 100$=20%

None of significant digit are at least correct in the estimate root of x=1.25 because approximate error is greater than 5%

More iteration conducted show in table

Iteration	a	b	X	F(xm)	$\lvert\epsilon\rvert$%	update
1	1	2	1.5	0.875		b=c
2	1	1.5	1.25	-0.2969		a=c
3	1.25	1.5	1.375	0.2246		b=c
4	1.25	1.375	1.3125	-0.0515		a=c
5	1.3125	1.375	1.3438	0.0826		b=c
6	1.3125	1.3438	1.3281	0.0146		b=c
7	1.3125	1.3281	1.3203	-0.0187		a=c
8	1.3203	1.3281	1.3242	-0.0021		a=c
9	1.3242	1.3281	1.3262	0.0062		b=c
10	1.3242	1.3262	1.3252	0.002		b=c
11	1.3242	1.3252	1.3247	0	0.0377	a=c

Approximation root=1.3247

No of significant digit at least correct is given by largest value for m

$|\epsilon|$<=0.5*10^2-m

0.0377<=0.5*18^2-m

log0.0754<=10^2-m

m<=2-1.1226

m=3

2.2 .REGULA FALSE METHOD

It is method for solving equation in one unknown. It is quite similar to bisection method algorithm. It was developed because bisection method converges slowly.

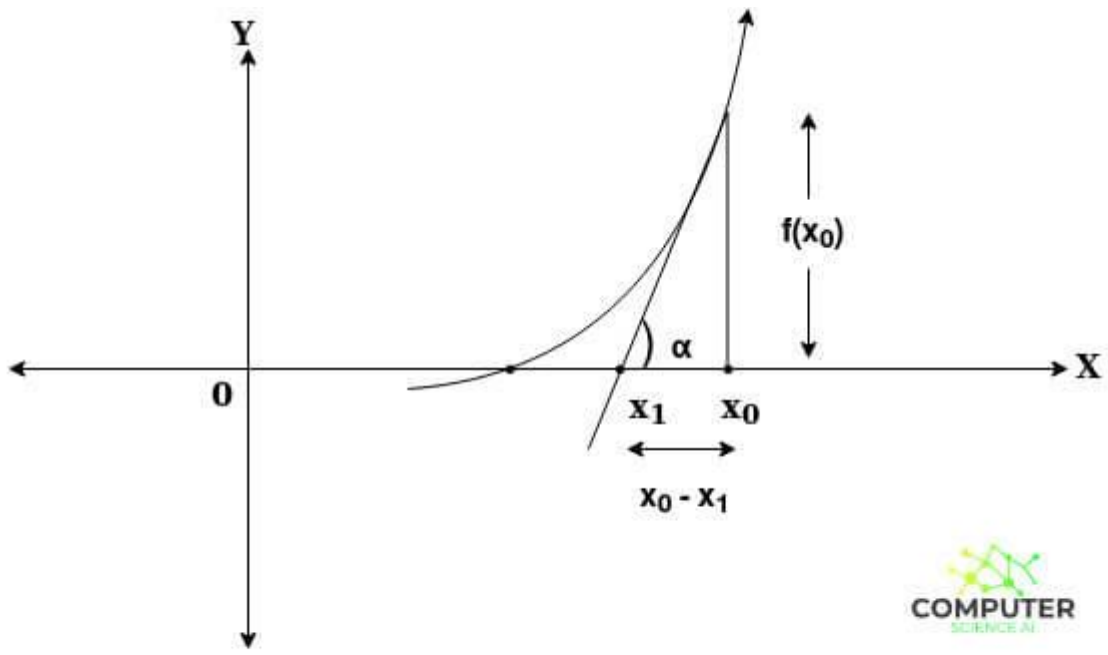


Fig. Derivation of Newton-Raphson's Formula Without Using Taylor's Expansion

ALGORITHM

- **Choose two guesses for root, such that f(a)f(b)<0**
- **Estimate root Xm=af(b)-bf(a)/f(b)-f(a)**
- **Find absolute error |error|=$|\frac{X_m^{new}-x_m^{old}}{x_m^{new}}|$*100**
- **If we get |error|<10^{-3} then stop however estimated root is 10^{-3}**

$$m = \frac{f(b) - f(a)}{b - a}$$

Advantages

1. It always converges.
2. It does not require the derivative.
3. It is a quick method.

Disadvantages

1. One of the interval definitions can get stuck.
2. It may slowdown in unfavourable situations

$f(x)=2x^2-2x-5$

Find a root of an equation $f(x)=2x3-2x-5$ using False Position method (regulafalsi method)

Solution:

Here $2x^2-2x-5=0$

Let $f(x)=2x^2-2x-5$

Here

x	0	1	2
f(x)	-5	-5	7

1st iteration :

Here $f(1)=-5<0$ and $f(2)=7>0$

∴ Now, Root lies between $x0=1$ and $x1=2$

$x2=x0-f(x0)\cdot x1-x0f(x1)-f(x0)$

$x2=1-(-5)\cdot 2-17-(-5)$

$x2=1.41667$

$f(x2)=f(1.41667)=2\cdot 1.416673-2\cdot 1.41667-5=-2.14699<0$

n	$x0$	$f(x0)$	$x1$	$f(x1)$	$x2$	$f(x2)$	Update
1	1	-5	2	7	1.41667	-2.14699	$x0=x2$
2	1.41667	-2.14699	2	7	1.55359	-0.60759	$x0=x2$
3	1.55359	-0.60759	2	7	1.58924	-0.15063	$x0=x2$
4	1.58924	-0.15063	2	7	1.59789	-0.0361	$x0=x2$
5	1.59789	-0.0361	2	7	1.59996	-0.00858	$x0=x2$
6	1.59996	-0.00858	2	7	1.60045	-0.00203	$x0=x2$
7	1.60045	-0.00203	2	7	1.60056	-0.00048	$x0=x2$

Codes:

```
defreg_falsi(f,x1,x2,tol=1.0e-6,maxfpos=100):# regula false method accept value from below

if f(x1) * f(x2)<0: # it shows if multiplication of function value at x1 and x2 is less than 0 then

preceed to values under this statement

forfposin range(1,maxfpos+1):

xh= x2 - (x2-x1)/(f(x2)-f(x1)) * f(x2) # finding root value

if abs(f(xh)) <tol: # if value of function at xh is less than tolerance then proceed to values that

is u der if statement#

break # # if above is true then terminate the loop

elif f(x1) * f(xh) < 0: # multiplication of function values is leass than zero then proceed to

values that is u der if statement

        x2 =xh # xh assign to x2

else:

        x1 =xh # if condition not true then xh assign to x1 ,now function go to to finding xh

else:

    print('No roots exists within the given interval') # f(x1) * f(xh) < 0 if not this then

compiler print this

returnxh, fpos

y =lambda x:2x3-2x-5  # function value

x1 = float(input('enter x1: ')) # input float x1 value from user

x2 = float(input('enter x2: ')) # input float x2 value from user

r, n =reg_falsi(y,x1,x2)

print('The root = %f at %d false position'%(r,n)) # give output of function r is root and n is

iteration number
```

2.3. Newton Raphson method

The Newton-Raphson method is a way to quickly find a good approximation for the root of a real-valued function f (x) = 0 f(x) = 0 f(x)=0. It uses the idea that a continuous and differentiable function can be approximated by a straight line tangent to it.

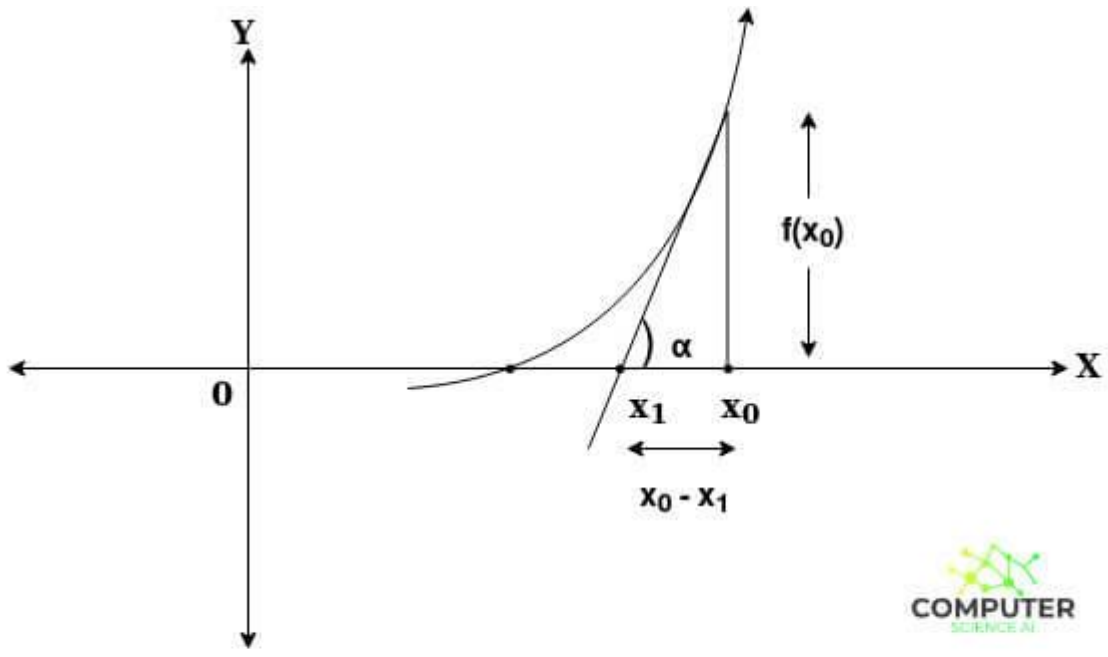


Fig. Derivation of Newton-Raphson's Formula Without Using Taylor's Expansion

Advantages

1. One of the fastest methods which converges to root quickly.
2. Converges on the root quadraticallyi.e rate of convergence
3. As we go near to root, number of significant digits approximately doubles with each step.

. It makes this method useful to get precise results for a root which was previously obtained from some other convergence method. 5. Easy to convert to multiple dimension.

DISADVANTAGES

- IT'S CONVERGENCE IS NOT GUARANTEED. ...
- DIVISION BY ZERO PROBLEMS CAN OCCUR.
- ROOT JUMPING MIGHT TAKE PLACE THEREBY NOT GETTING INTENDED SOLUTION.
- INFLECTION POINT ISSUE MIGHT OCCUR.
- SYMBOLIC DERIVATIVE IS REQUIRED.
- IN CASE OF MULTIPLE ROOTS, THIS METHOD CONVERGES SLOWLY.

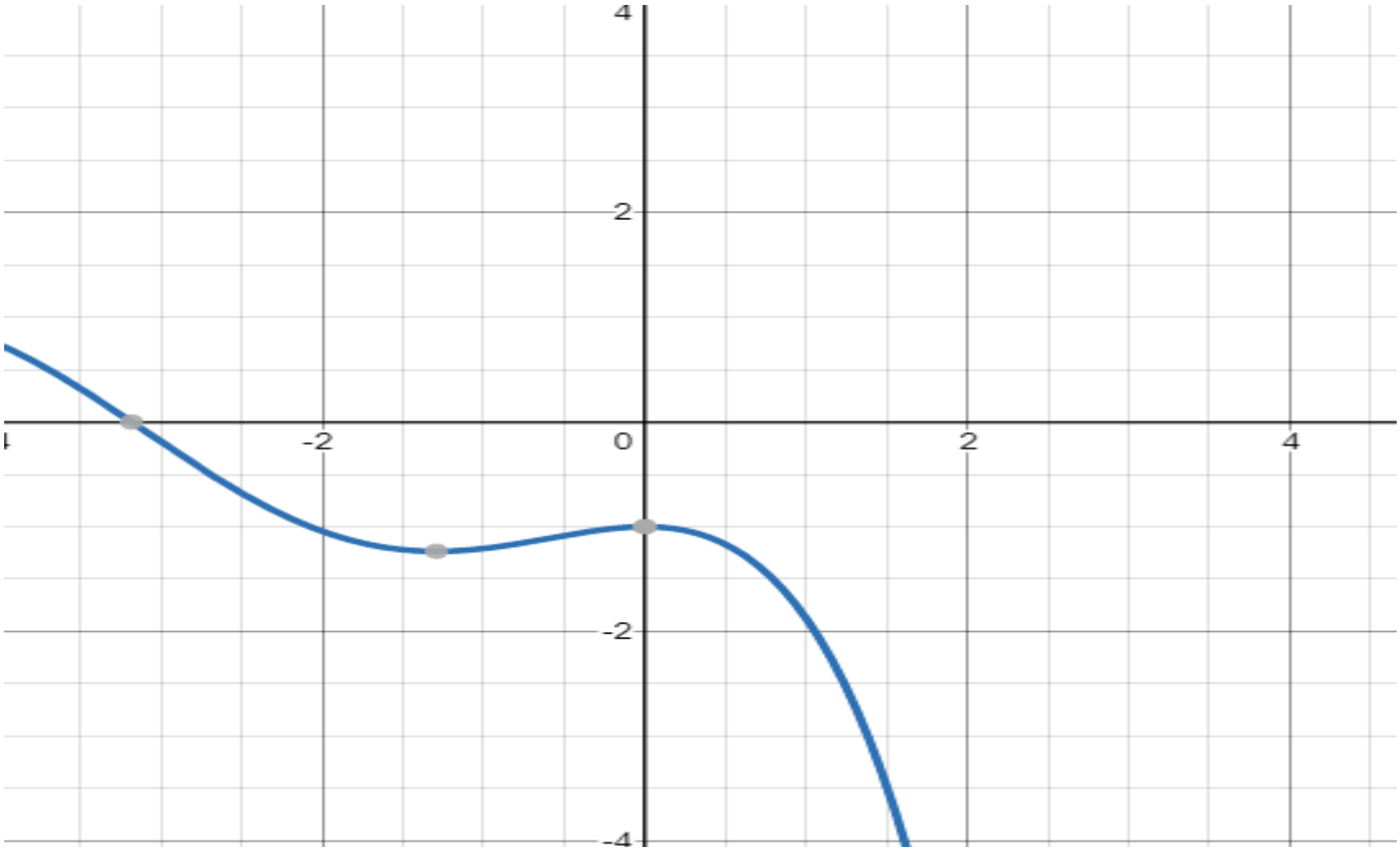
ALGORITHM

- WE HAVE A FUNCTION
- TAKING DERIVATIVES OF THE FUNCTION
- REMEMBER DERIVATIVE MUST NOT $=0$
- OUR ROOT IS ALWAYS IN $+$VE AND $-$VE POINT
- WE TAKE ONLY ONE INITIAL GUESS BETWEEN THESE POINTS
- FORMULA:$X_{N+1}=X_N$-F (XN) \F'(XN)

X NEW WILL BE THE ASSIGN AS X_N

we have a function f(x)=$sinx - e^{-x}$

Graph



f(x)= $sinx - e^{-x}$

Taking derivative

f'(x) =cosx + e^{-x}

we have formula of **Newton Raphson Method**

$x_{n+1}=x_n - \frac{f(x)}{f'(x)}$

x=0

f(0)= $sin0 - e^{-0}$=-1

x=

f(1) = $sin1 - e^{-1}$=0.4736

so our root is between -1 and 0.4736

x_{n+1}= $Xn - \frac{sinA - e^{-x}}{cosA + e^{-x}}$

X_0=0.1

n=0

$x_1 = Xn - \frac{sinx_n - e^{-x_n}}{cosx_n + e^{-x_n}}$

X1=0.5237

Usingcalculator X=$A - \frac{sinA - e^{-A}}{cosA + e^{-A}}$

Roots of function is 0.5885 up to four decimal repeated

Iteration	n	xn	x_{n+1}=$x_n - \frac{sinA - e^{-x}}{cosA + e^{-x}}$
1	0	x_1	X1=0.5237
2	1	x_2	x2=0.5869
3	2	x_3	X3=0.5885
4	3	x_4	X4=0.5885

Code:

```
from math import sin

defnewton(fn,dfn,x,tol,maxiter

foriin range(maxiter)

xnew= x -fn(x)/dfn(x)

if abs(xnew-x)<tol:# if error is leass than tolerance

break

    x =xnewsinx-e^x

returnxnew,


y =lambda x:   # this is function
```

value

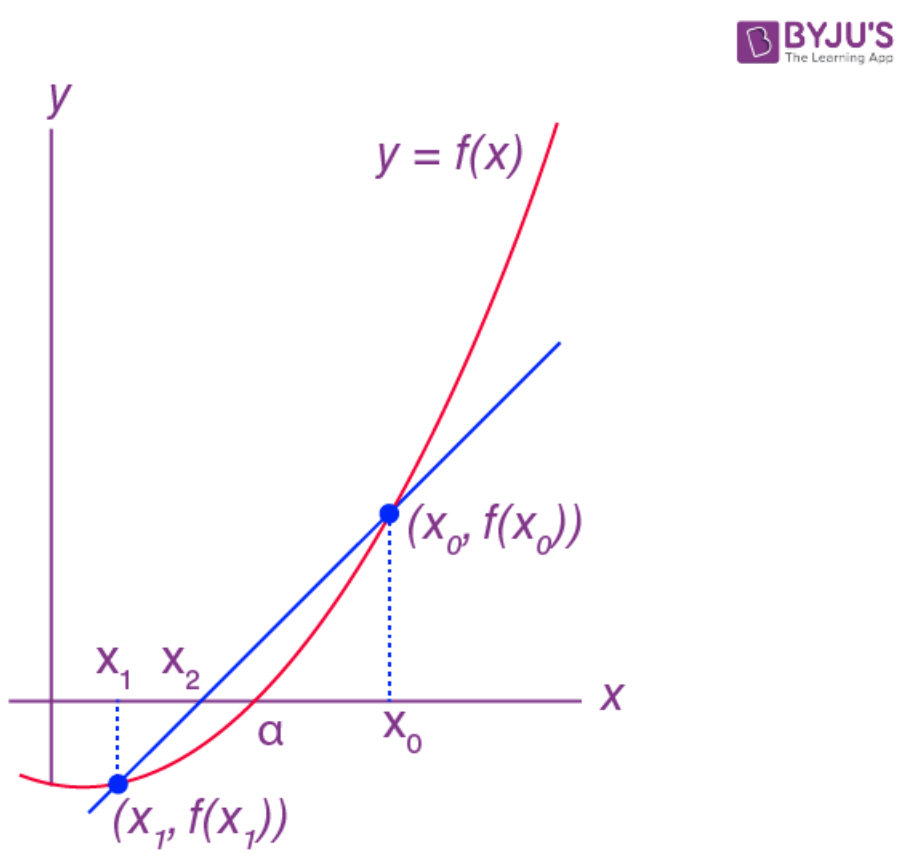
dy=**lambda** x : cosx-e^{-x}

x, n = newton(y, dy, 5, 0.0001, 100

print('the root is %.3f at %d iterations.'%(x,n)) # print root is x and at iteration is n

2.4 Secant Method

The secant method is a root-finding algorithm that uses a succession of roots of secant lines to better approximate a root of a function f. The secant method can be thought of as a finite-difference approximation of Newton's method.



Advantages

The secant method has the following advantages:

- It converges quicker than a linear rate, making it more convergent than the bisection method.
- It does not necessitate the usage of the function's derivative, which is not available in a number of applications.
- Unlike Newton's technique, which requires two function evaluations in every iteration, it only requires one.
-

Disadvantages

The secant method has the following drawbacks:

- The secant method may not converge.
- The computed iterates have no guaranteed error bounds.
- If $f_0(\alpha) = 0$, it is likely to be challenging. This means that when $x = \alpha$, the x-axis is tangent to the graph of $y = f(x)$.
- Newton's approach is more easily generalized to new ways for solving nonlinear simultaneous systems of equations.

ALGORITHM

- WE HAVE A FUNCTION F(X).
- WE HAVE TO FIND ROOT .
- OUR ROOT IS ALWAYS IN +VE AND −VE POINT
- WE TAKE ONLY ONE INITIAL GUESS BETWEEN THESE POINTS

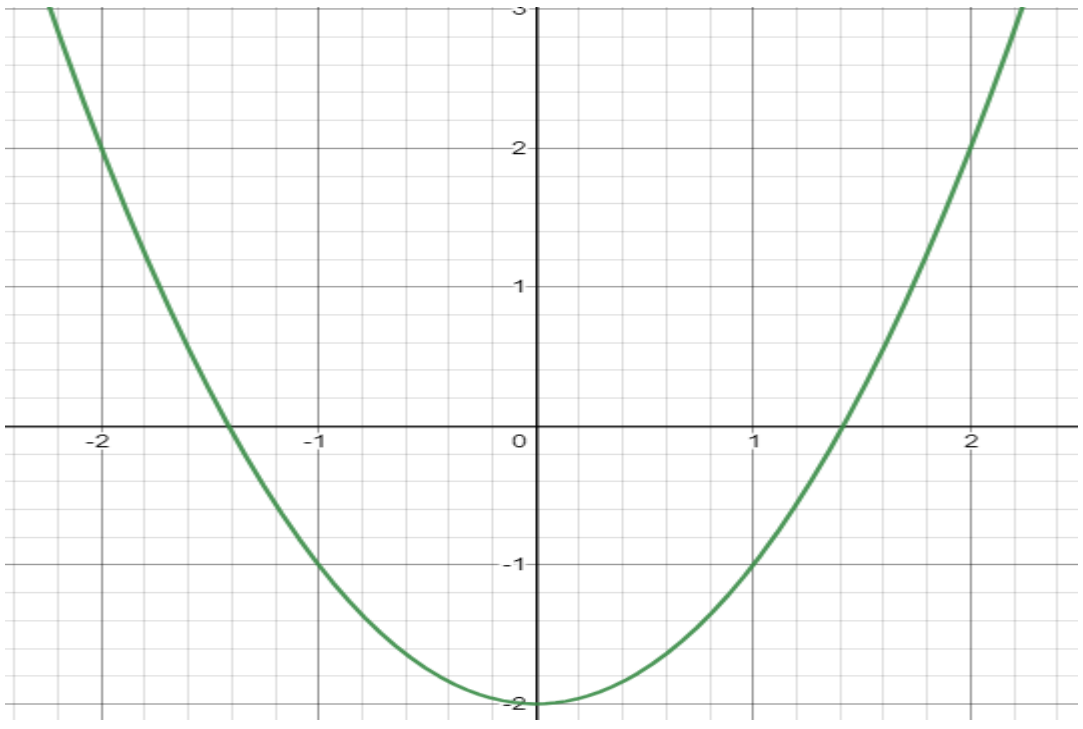
➢ Formula : $\frac{af(b)-bf(a)}{f(b)-f(a)}$

Remember denominator should not equal to zero

LET WE HAVE AN EXAMPLE

$$[f(x) = x^2 - 2]$$

Graph



$$[f(x) = x^2 - 2]$$

F(1)=-2

F(2)=0

[a,b]=[1,2]

Formula: $\frac{af(b)-bf(a)}{f(b)-f(a)}$

C=a($b^2 - 2$)-b($a^2 - 2$)\ $b^2 - 2 - a^2 + 2$

TABLE

a	b	c	F(c)
1	2	1.3333	-0.2223
2	1.3333	1.4000	-0.040
1.3333	1.4000	1.4146	-0.0010
1.4000	1.4146	1.4142	0.0000
1.4146	1.4142	1.4142	0.0000

code

```
from math import sin

def secant(fn,x1,x2,tol,maxiter): # secant method take values of function, x1 , x2 ,tolerance and maxitor
foriin range(maxiter):

xnew= x2 - (x2-x1)/(fn(x2)-fn(x1))*fn(x2) # this will calculate root and assign to xnew
if abs(xnew-x2) <tol:

break
else:
        x1 = x2

x2=xnew
```

```
else:
    print('warning: Maximum number of iterations is reached')
returnxnew,
i

f =lambda x: x²-2


x1= float(input('enter x1: '))
x2= float(input('enter x2: '))


r, n = secant(f,x1,x2,1.0e-6,100)
print('Root = %f at %d iterations'%(r,n)) # print root and iteration
```

2.5 Fixed point method

The **fixed point iteration** method in numerical analysis is used to find an approximate solution to algebraic and transcendental equations. The fixed point iteration method uses the concept of a fixed point in a repeated manner to compute the solution of the given equation. A fixed point is a point in the domain of a function g such that g(x) = x. In the fixed point iteration method, the given function is algebraically converted in the form of g(x) = x.

Advantages

1. Fast convergence: It converges fast, if it converges. Which means, in most cases we get root (answer) in less number of steps.
2. It requires only one guess.
3. Formulation of this method is simple. So, it is very easy to apply.
4. It has simple formula so it is easy to program.
5. Derivation is more intuitive, which means it is easier to understand its behaviour, when it is likely to converge and when it is likely to diverge

ALGORITHM

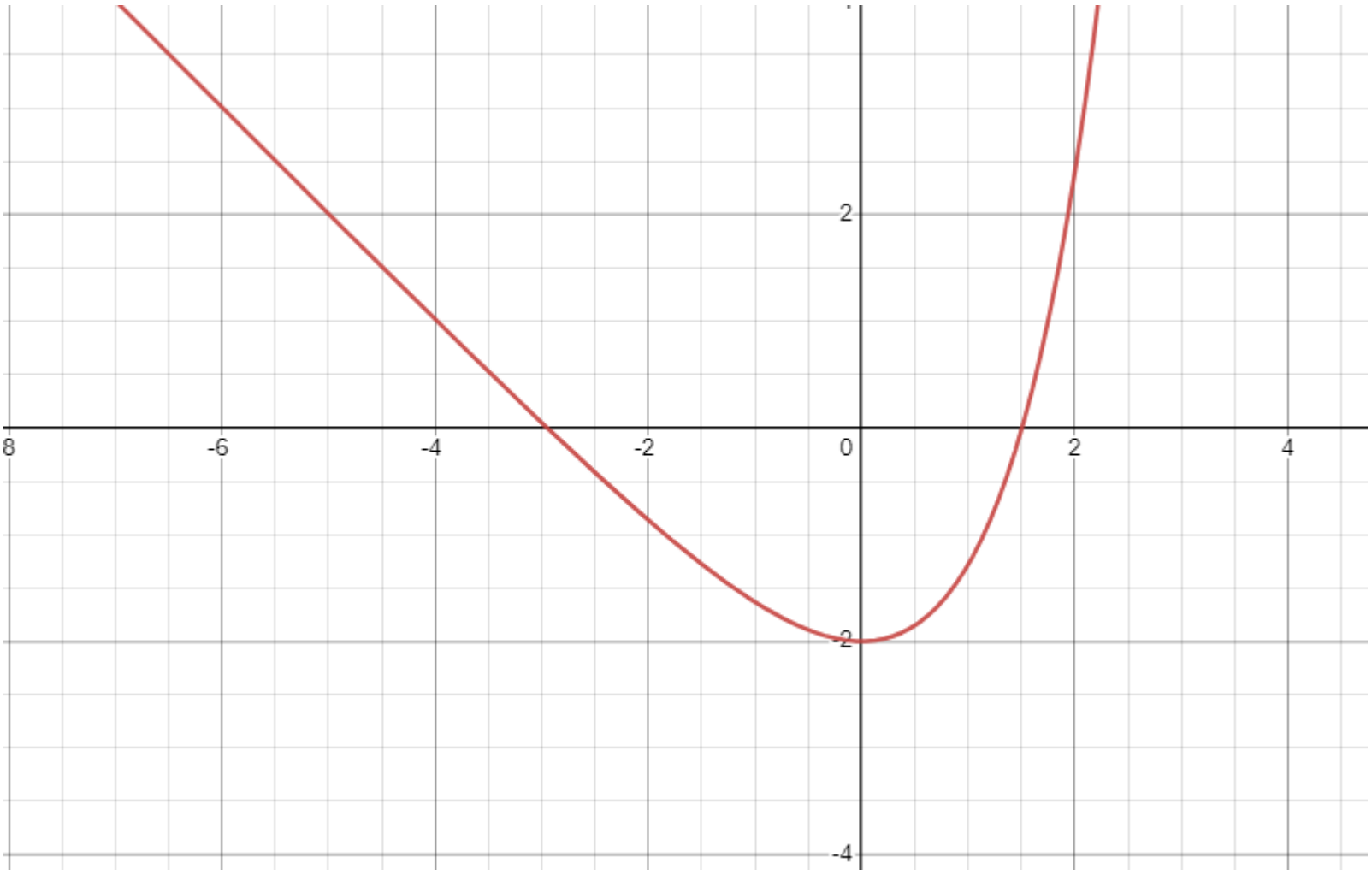
- **Given an equation f(x) = 0**
- **Convert f(x) = 0 into the form x = g(x)**
- **Let the initial guess be x0**
- **xi+1= g(xi) i=1,2,3,…**

RESTRICTION

f(a)f(b) <0
$|g'(x_0)|$ <1

We understand through example

$$f(x) = e^x - x - 3$$



- $f(-3) = 0.0497$

  $f(-2) = -0.8646$

- $f(-3)f(-2) < 0$

Points are [-3,-2]

Take $x_0 = -3$

- Decompose the function

We will write the equation in the $x = g(x)$

$$e^x - x - 3 = 0$$

$$x = e^x - 3$$

x= g(x)

- take derivative g’(x)=e^x
- |g’(-3)|=|e^{-3}|=0.4997 <1

Making formula for function

- xi+1= g(xi)

$$x_i = e^{x_{1-1}} - 3$$

$$x_1 = e^{x_0} - 3$$

$$x_0 = -3$$

i=1

$$x_1 = -2.9502$$

f(x_1)=0.00254

Table

iteration	**i**	x_i	$f(x_i)$
1	1	-2.9502	0.00254
2	2	-2.9476	0.000034

System of linear equation

In the system if linear equations we will solve Jacobi and Guass Siedal and SOR methods and their convergence criteria.

Before this we should know the concept of norms

The I1 norm is given by $||X||1 = \sum_{i=1}^{n} |x|$

The I2 normor Euclidean norm is given by $||x||2 = \sum_{i=2}^{n} |x|$

The Iinf norm or max norm is given by $|x||inf = \max_{0 \le x \le 1} |x|$

Convergence Criteria

A sequence of vectors $\prod_{k=1}^{inf}$ Xk inRnis said to converge to x with respect to norm ||.|| if given any $\varepsilon > 0$ and there exists an integer N (ε) such that

$||xk - x|| < \varepsilon$ for all $k \geq N$ (ε)

Jacobi Method

ALGORITHM

Set k=1

While k≤N do step 3-6

For i=1, 2…n

$Xi = [\sum_{j=1}^{n}(-a\ XO) + bi$

If |x-Xo|<Tol then stop

K=k+1

Then result x1, x2,….

Stopping criteria for Jacobi is

Example

4x+y-z=5

-x+3y+z=-4

2x+2y+5z=1

First find system is strickly dominant

|4|>|1|+|-1|

|3|>|-1|+|1|

|5|>|2|+|2|

Set iteration

$x_{k+1}=\frac{1}{4}$(5-yk+zk)

$y_{K+1}=\frac{1}{3}$(-4+xk-zk)

$Z_{K+1}=\frac{1}{5}$(1-2yk-2zk)

Take initial guess x=0,y=0,z=0

1st iteration

$x_1=\frac{1}{4}(5+0)=1.25$

$x_2=\frac{1}{3}(-4+0-0)=-1.33$

$x_3=\frac{1}{5}(1-2(0)-2(0))=0.22$

Table

N	x	y	z
1	1.25	-1.3333	0.2
2	1.6333	-0.9833	0.2333
3	1.5542	-0.8667	-0.06
4	1.4517	-0.7953	-0.075
5	1.4301	-0.8244	-0.0626
6	1.4405	-0.8358	-0.0422
7	1.4484	-0.8391	-0.0419
8	1.4482	-0.8357	-0.0451
9	1.4476	-0.8356	-0.045

While implementing Jacobi method then it is not that the the matrix alwaysstrickly dominant then we go for another method we find spectral radius.So for this spectral radius<1.

Spectral radiusthe spectral radius for a square matrix is defined simply as the largest absolute value of its eigenvalue.

We first do D-1(L+U)Xn+D-1b

D^{-1} (L+U)=T

$$D^{-1}=\begin{bmatrix} \frac{1}{VALUE\ OF\ THIS} & 0 & 0 \\ 0 & \frac{1}{VALUE\ OF\ THIS} & 0 \\ 0 & 0 & \frac{1}{VALUE\ OF\ THIS} \end{bmatrix}$$

L=lower triangular matrix

U=upper triangular matrix

After finding T we multiply it with lambda and calculateEigen value and we will get all Eigen values and they are less than 1.

P(Tj)<1

GuassSiedal Method

in this method we use the new value x(k+1) as soon as they are known

Its algorithm just like Jacobi but we use update values in each step of iteration .

Example

4x+y-z=5

-x+3y+z=4

2x+2y+5z=1

Solution

$x_{k+1}=\frac{1}{4}$(5-yk+zk)

$y_{K+1}=\frac{1}{3}$(4+xk+1-zk)

$Z_{K+1}=\frac{1}{5}$(1-2xk+1-2yk+1)

Initial guesses are (0, 0, 0)

1st iteration

$X_1 = \frac{1}{4}(5\text{-}0\text{-}0) = 1.25$

$Y_1 = \frac{1}{3}(4\text{-}1.25\text{-}0) = 1.75$

$Z_1 = \frac{1}{5}(1\text{-}2(1.25)\text{-}2(1.75)) = -1$

Iteration	x	y	z
1	1.25	1.75	-1
2	0.5625	1.8542	-0.7667
3	0.5948	1.7872	-0.7528
4	0.615	1.7839	-0.7617
5	0.6123	1.7913	-0.7614
6	0.6118	1.7911	-0.7612

While implementing guasssiedal we may face a case in which diagonal is dominant. So may interchange rows and if from row interchange we also don't get diagonal dominant then we go for spectral radius

Spectral Radius

The spectral radius for a square matrix is defined simply as the largest absolute value of its Eigen value.

T (g) $= (D - L)^{-1}$U

Then we will find Eigen values

Successive over-relaxation

$3x-y+z=-1$

$-x+3y-z=7$

$x-y+3z=-7$

Solve Equations 3x-y+z=-1,-x+3y-z=7,x-y+3z=-7 using SOR (Successive over-relaxation) method

Solution:

We know that, for symmetric positive definite matrix the SOR method converges for values of the relaxation parameter w from the interval $\mathbf{0<w<2}$

The iterations of the SOR method

1. Total Equations are 3

3x-y+z=-1

-x+3y-z=7

x-y+3z=-7

2. From the above equations, First write down the equations for Gauss Seidel method

$X_{K+1}=\frac{1}{3}(-1+Y_K-Z_K)$

$Y_{K+1}=\frac{1}{3}(7+X_{K+1}+Z_K)$

$Z_{K+1}=\frac{1}{3}(-7-X_{K+1}+Y_{K+1})$

3. Now multiply the right hand side by the parameter *w* and add to it the vector *xk* from the previous iteration multiplied by the factor of (1-*w*)

$xk+1=(1-w)\cdot xk+w\cdot\frac{1}{3}(-1+yk-zk)$

$yk+1=(1-w)\cdot yk+w\cdot\frac{1}{3}(7+xk+1+zk)$

$zk+1=(1-w)\cdot zk+w\cdot\frac{1}{3}(-7-xk+1+yk+1)$

4. Initial gauss (x,*y*,z)=(0,0,0) and *w*=1.25

Solution steps are

1*st* Approximation

$x1=(1-1.25)\cdot 0+1.25\cdot 13[-1+(0)-(0)]=(-0.25)\cdot 0+1.25\cdot 13[-1]=0\pm 0.41667=-0.41667$

$y1=(1-1.25)\cdot 0+1.25\cdot 13[7+(-0.41667)+(0)]=(-0.25)\cdot 0+1.25\cdot 13[6.58333]=0+2.74306=2.74306$

$z1=(1-1.25)\cdot 0+1.25\cdot 13[-7-(-0.41667)+(2.74306)]=(-0.25)\cdot 0+1.25\cdot 13[-3.84028]=0\pm 1.60012=-1.60012$

ITTERATION	**X**	**Y**	**Z**
1	-0.41667	2.74306	-1.60012
2	1.49715	2.188	-2.22878
3	1.04937	1.84824	-2.01411
4	0.9428	2.00073	-1.97234
5	1.00308	2.01263	-2.00294
6	1.00572	1.998	-2.00248
7	0.99877	1.99895	-1.9993
8	0.99958	2.00038	-1.99984
9	1.0002	200005	-2..0001

Finite differences

A finite differences are basically a table for forward, backward and central differences. So after generating table we go for interpolation. We will use all these in interpolation.

Interpolation

It is an estimation if values with in two known values in a sequence if value.

Newton forward Difference

Newton's forward interpolation is a polynomial interpolation that is based on Newton's forward operator's initial value and degrees. The amount of data points is one less than the degree of polynomial fitted. This form of interpolation is only utilised when the data points are evenly spaced.

Find Solution using Newton's Forward Difference formula

x	f(x)
0	1
1	0
2	1
3	10

x = -1

Solution:

The value of table for x and y

X	0	1	2	3
y	1	0	1	10

Newton's forward difference interpolation method to find solution

Newton's forward difference table is

X	y	Δy	$\Delta^2 y$	$\Delta^3 y$
0	1	-1	2	6
1	0	1	8	
2	1	9		
3	10			

The value of x at you want to find the f(x):x=-1

h=x1-x0=1-0=1

p=x-x0h=-1-01=-1

Newton's forward difference interpolation formula is

y(x)=y0+pΔy0+p(p-1)\2!·Δ2y0+p(p-1)(p-2)\3!·Δ3y0

y(-1)=1+(-1)×-1+-1(-1-1)\2×2+-1(-1-1)(-1-2)\6×6

y(-1)=1+1+2-6

y(-1)=-2

Solution of newton's forward interpolation method y(-1)=-2

code

```
adff1= (f(x+h)-f(x))/h

dff2= (f(x+2*h)-2*f(x+h)+f(x))/h**2

print("ffd\t% f\t% f\t% f\t% f"%(dff1,dff1-df1,dff2,dff2-df2))
```

Newton Backward Difference

In order to reduce the number of numerical computations required to compute a large number of interpolated values using the existing interpolation formula,

Find Solution of an equation x^3-x+1 using Newton's Backward Difference formula
x1 = 2 and x2 = 4
x = 3.75
Step value (h) = 0.5
Finding f(2)

Solution:
Equation is f(x)=x3-x+1.

The value of table for x and y

x	2	2.5	3	3.5	4
y	7	14.125	25	40.375	61

Newton's backward difference interpolation method to find solution

Newton's backward difference table is

X	y	Δy	$\Delta^2 y$	Δ^3	$\Delta^4 y$
2	7				
2.5	14.125	7.125			
3	25	10.875	3.75		
3.5	40.375	15.375	4.5	0.75	
4	61	20.625	5.25	0.75	0

The value of x at you want to find the f(x):x=3.75

h=x1-x0=2.5-2=0.5

p=x-xnh=3.75-40.5=-0.5

Newton's backward difference interpolation formula is
y(x)=yn+p∇yn+p(p+1)2!·∇2yn+p(p+1)(p+2)3!·∇3yn+p(p+1)(p+2)(p+3)4!·∇4yn

y(3.75)=61+(-0.5)×20.625+-0.5(-0.5+1)2×5.25+-0.5(-0.5+1)(-0.5+2)6×0.75+-0.5(-0.5+1)(-0.5+2)(-0.5+3)24×0

y(3.75)=61-10.3125-0.6562-0.0469+0

y(3.75)=49.9844

Solution of newton's backward interpolation method y(3.75)=49.9844

Code

dff1= (f(x+h)-f(x))/h

dff2= (f(x+2*h)-2*f(x+h)+f(x))/h**2

print("ffd\t% f\t% f\t% f\t% f"%(dff1,dff1-df1,dff2,dff2-df2))

Langrane interpolation

The Lagrange interpolating polynomial is the unique polynomial of
lowest degree that interpolates a given set of data.

Find Solution using Lagrange's Interpolation formula

x	f(x)
2	0.69315
2.5	0.91629
3	1.09861

x = 2.7

Solution

x	f(x)
2	0.69315
2.5	0.91629
3	1.09861

:
The value of table for x and y

Lagrange's Interpolating Polynomial
The value of x at you want to find Pn(x):x=2.7

Lagrange's formula is
f(x)=(x-x1)(x-x2)\(x0-x1)(x0-x2)×y0+(x-x0)(x-x2)\(x1-x0)(x1-x2)×y1+

(x-x0)(x-x1)\(x2-x0)(x2-x1)×y2

y(2.7)=(2.7-2.5)(2.7-3\)(2-2.5)(2-3)×0.69315+(2.7-2)(2.7-3)\(2.5-2)(2.5-3)×0.91629+

(2.7-2)(2.7-2.5)\(3-2)(3-2.5)×1.09861

y(2.7)=(0.2)(-0.3)\(-0.5)(-1)×0.69315+(0.7)(-0.3)\(0.5)(-

0.5)×0.91629+(0.7)(0.2)\(1)(0.5)×1.09861

y(2.7)=-0.060.5×0.69315+-0.21-0.25×0.91629+0.140.5×1.09861

y(2.7)=0.994116

Solution of the polynomial at point 2.7 is y(2.7)=0.99411

Newton's divided difference interpolation formula

Is an interpolation technique used when interval difference is not same for all sequence of value.Suppose $f(x_0),f(x_1),f(x_2)\ldots\ldots f(x_n)$ be the n+1 values of function y=f(x) corresponding to the arguments $x=x_0,x_1,x_{2\ldots\ldots}x_n$ where interval difference are not same.

X	**Y=f(x)**	$\Delta \boldsymbol{f}(\boldsymbol{x})$	$\Delta^{\boldsymbol{2}} \boldsymbol{f}(\boldsymbol{x})$
x_0	y_0	$\frac{y_1-y_0}{x_1-x0}=F[x_0,x_1]$	
			$\frac{f[x_1,x_2]-f[x_0,x_1]}{x_2-x_0}$
x_1	y_1	$\frac{y_2-y1}{x_2-x1}=f[x_1,x_2]$	

Now from example it will be more clear to you

Question

Using newton divided formula find f(4)

X	4	5	7	10	11	13
y=f(x)	48	100	294	900	1210	2028

Now we we will go towards solutions

X	F(x)	$\Delta f(x)$	$\Delta^2 f(x)$	$\Delta^3 f(x)$	$\Delta^4 f(x)$
4	48				
		$\frac{100-48}{5-4}=52$			
5	100		$\frac{97-52}{7-4}=15$		
		$\frac{294-100}{7-5}=97$		1	
7	294		$\frac{202-97}{10-5}=21$		0
		$\frac{900-294}{10-7}=202$		1	
10	900		$\frac{310-202}{11-7}=27$		0
		$\frac{1210-900}{11-10}=310$		1	
11	1210		$\frac{409-310}{13-10}=33$		
		$\frac{2028-1210}{13-11}=409$			
13	2028				

$F(x)=f(x_0)+(x-x_0)\Delta f(x_0)+(x-x_0)(x-x_1)\Delta^2 f(x_0)+(x-x_0)(x-x_1)(x-x_2)\Delta^3 f(x_0)$……..

$F(8)=48+(8-4)52+(8-4)(8-5)15+(8-4)(8-5)(8-7)1+0=210$

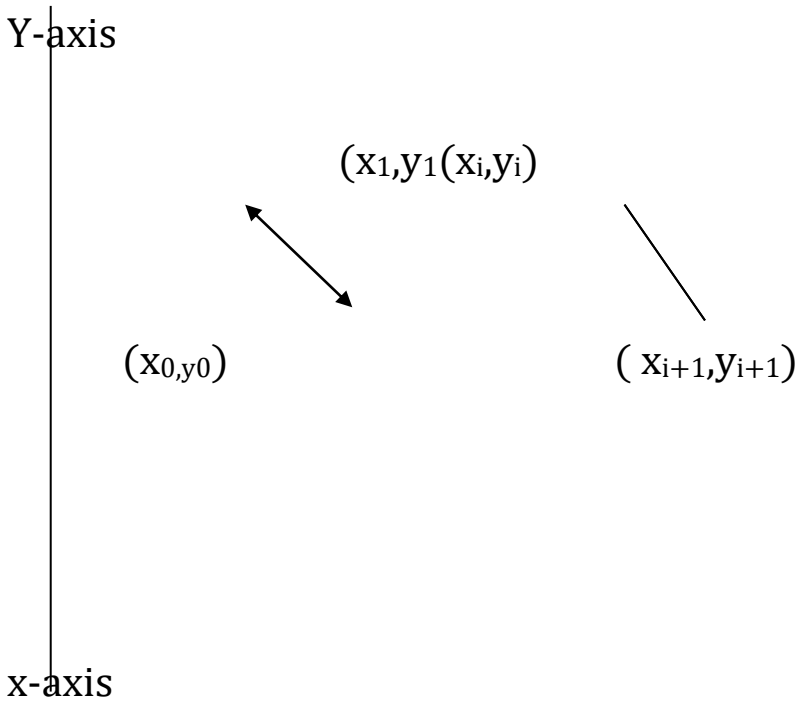
Spline interpolation

Spline interpolation divided into 3 types

- linear spline
- quardatic spline
- cubic spline

Linear spline interpolation

Given (x_0,y_0),…….(x_n,y_n) interpolate data to linear spline. Values of x should be in particular order like $x_0<x_1<\ldots\ldots x_n$



$$F(x)=y_i+\frac{Y_{i+1}-Y_i}{x_{i+1}-x_i}(x-x_i)$$

Draw back

- It is not able to use information from other data points .It depends on particular data points.
- Derivatives is not continuous , even first derivative in continuous.

Question

Find value of velocity at 16 by using linear spline interpolation formula?

Time	velocity
0	0
10	227.04
20	517.35
15	362.78
22.5	602.97

First we arrange in order

Time	velocity
0	0
10	227.04
15	362.78
20	517.35
22.5	602.97

Now we have to find v(16) lie in interval [15,20], so

$F(16)=y_{15}+\frac{y_{20}-y_{15}}{20-15}(x\text{-}15)$

$F(16)=362.78+\frac{517.35-362.78}{20-15}(16\text{-}15)$

$F(16)=393.7$ m/sec

Quadratic spline

It is type of spline interpolation where every point connecting two consecutive points is a quadratic function. Thus for data n+1 pair of points, n spline function are needed. Quadratic splines are

$Y_1=a_1x^2+b_1x+c_1 \quad x_0<x<x_1$

$Y_2=a_2x^2+b_2x+c_2 \quad x_1<x<x_2$

$Y_n=a_nx^2+b_nx+c_n \quad x_{n-1}<x<x_n$

There are three unknown coefficients in every quadratic spline (a_1,b_1,c_1). Thus there is needed of 3n linear equation derived in the following steps.

1) All given points are part of spline interpolation

   There is total of 2n linear equations derived on this step

2) The first derivative of two spline sharing an intermediate points are equal.

   There is total of n-1 linear equation derived on this step

3) The second derivative of 1st spline spline is assumed to be zero.

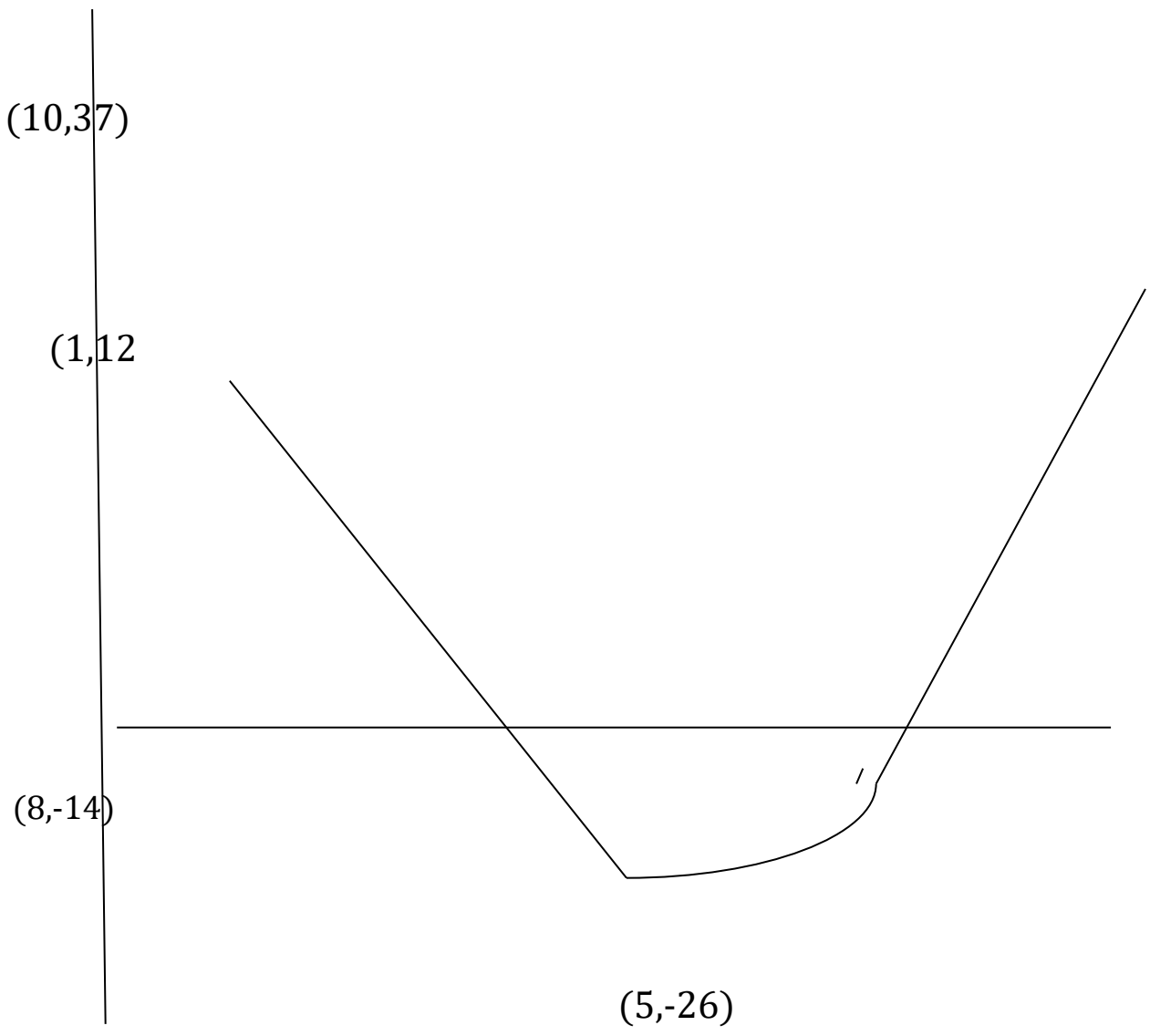
There is only one equation derived on this step that is $2a_1=0$

Because of this assumption the first spline is actually a line.

Question

Find f(7) for given set of data below using quadratic spline interpolation.

X	F(x)
1	12
5	-26
8	-14
10	37



No of equations will be $3(3)=9$

The given set has 4 set of points so .Thus there is need of n=4-1=3 quadratic function to perform interpolation.

$Y_1=a_1x^2+b_1x+c_1 1<x<5$

$Y_2=a_2x^2+b_2x+c_2 \quad 5<x<8$

$Y_{3=}a_3x^2+b_3x_+c_3 \quad 8<x<10$

The system has 3n =3(3)=9 number of unknown . 9 equations needed to solve these unknown.

ALGORITHM

1) 2n=2(3)=6

Point (1,12) is only part of the 1st quadratic spline

$12=a_1(1)^2+b(1)+c_1$

$a_{1+}b_{1+}c_{1=}12$ ……..eq 1

Point (5,-26) is part of 1st and 2nd quadratic spline.

$-26=a_125+5b_1+c_1$ ……..eq2

$-26=a_225+5b_2+c_2$……..eq3

Points (8,-14) is part of 2nd and 3rd spline

$-14=64a_2+8b_2+c_2$ ……..eq4

$-14=64a_3+8b_3+c_3$ ……..eq5

Point (10,37) is part of 3rd and 4rth spline

$37=100a_3+10b_3+c_3$……..eq6

2) 1 st derivatives of quadratic spline

The first interior point is (5,-26) first and second spline connected to it

$2a_1x+b_1=2a_2x+b_2$

$2a_1x+b_1-2a_2x+b_{2=0}$

$2a_1(5)+b_1-2a_2(5)+b_2=0$

$10a_1+b_1-10a_2+b_2=0$ ……eq7

The other interior point is (8,-14) and 2nd and 3rd spline connected to it

$2a_2x+b_2=2a_3x+b_3$

$2a_2x+b_2-2a_3x+b_{3=0}$

$2a_2(8)+b_1-2a_3(8)+b_3=0$

$16a_1+b_1-16a_2+b_2=0$ ……eq8

3) The second derivative is assumed to be zero

$2a_{1=0}$

$a_{1=0}$ ……eq9

Now

$$\begin{matrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 25 & 5 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 25 & 5 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 64 & 8 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 64 & 8 & 1= \\ 0 & 0 & 0 & 0 & 0 & 0 & 100 & 10 & 1 \\ 10 & 1 & 0 & -10 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 16 & 1 & 0 & -16 & -1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{matrix} \begin{bmatrix} 12 \\ -26 \\ -26 \\ -14 \\ -14 \\ 37 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Cubic spline

The fitting of a polynomial curve to a set of data points has applications in CAD(computer-assisted design), CAM (computer-assisted manufacturing), and computer graphics systems. An

operator wants to draw a smooth curve through data points that are not subject to error. Traditionally, it was common to use a French curve or an architect's spline and subjectively draw a curve that looks smooth when viewed by the eye. Mathematically, it is possible to construct cubic functions $S_k(x)$ on each interval $[x_k, x_{k+1}]$ so that the resulting piecewise curve $y = S(x)$ and its first and second derivatives are all continuous on the larger interval $[x_0, x_N]$. The continuity of S'(x)

Suppose that $\{(x_k, y_k)\}_{K=0}^{N}=0$ are N + 1 points, where

$$a = x_0 < x_1 < \cdots < x_N = b.$$

The function S(x) is called a cubic spline if there exist N cubic

polynomials Sk (x) with coefficients sk,0, sk,1, sk,2, and sk,3 that satisfy the following

properties:

I. $S(x) = S_k(x) = s_k,0 + s_k,1(x - x_k) + s_k,2(x - x_k)2 + s_k,3(x - x_k)3$

for $x \in [x_k, x_{k+1}]$ and $k = 0, 1, \ldots, N - 1$.

II. $S(x_k) = y_k$ for $k = 0, 1, \ldots, N$.

III. $S_k(x_{k+1}) = S_{k+1}(x_{k+1})$ for $k = 0, 1, \ldots, N - 2$.

IV. $S'_K = (x_{k+1})$ for $k = 0, 1, \ldots, N - 2$.

V. $S''_K(x_K+1) = S''_{K+1}(x_{k+1})$ for $k = 0, 1, \ldots, N - 2$.

$a_1x^3+b_1x^2+c_1x+d_1 \quad x_0 \le x \le x_1$ (1)

$a_2x^3+b_2x^2+c_2x+d_2 \quad x_1 \le x \le x_2$ (2)

⋮

$a_nx^3+b_nx^2+c_nx+d_n \quad x_{n-1} \le x \le x_n$ (3)

a_i;1, 2,…,n

b_i;1, 2,…,n } 4n constants/unknown

c_i;1, 2,…,n

d_i ;1,2,…,n

we have 4n constants so we should have 4n equations, now we will find all this 4n equations

Each cubic goes through 2 consecutives data points

from equation (1) we have

$y_0= a_1x_0^3+b_1x_0^2+c_1x_0+d_1$

$y_1= a_1x_1^3+b_1x_1^2+c_1x_1+d_1$

from equation (2) we have

$y_1= a_2x_1^3+b_2x_1^2+c_2x_1+d_2$

$y_2= a_2x_2^3+b_2x_2^2+c_2x_2+d_2$

⋮

from equation (3) we have

$Y_{n-1}= a_n x_{n-1}^3+b_n x_{n-1}^2+c_n x_{n-1}+d_{n-1}$

$Y_n= a_n x_n^3+b_n x_n^2+c_n x_n+d_n$

So we have find "2n" equations, now we will find remaining "2n" equation

Removing x_0 we will have n points (remaining) and removing x_1 we will have n-1 points (remaining) so we will have **n-1 interior points from n+1 points** (2 points x_0 , $x_n$are exterior points which doesn't satisfy the slopes are continuous because there is no cubic before x_0 and after x_n)

First derivative at two consecutive cubic are continuous at common interior points

$\frac{d}{dx}(a_1x^3+b_1x^2+c_1x+d_1) \Big|_{x=x_1} = \frac{d}{dx}(a_2x^3+b_2x^2+c_2x+d_2) \Big|_{x=x_1}$

$(3a_1x^2+2b_1x+c_1) \Big|_{x=x_1} = (3a_2x^2+2b_2x+c_2) \Big|_{x=x_1}$

$(3a_1x_1^2+2b_1x_1+c_1) = (3a_2x_1^2+2b_2x_1+c_2)$

$3a_1x_1^2+2b_1x_1+c_1-3a_2x_1^2-2b_2x_1-c_2=0$

$(3a_1-3a_2)x_1^2+(2b_1-2b_2)\,x_1+c_1-c_2=0$

$\frac{d}{dx}(a_{n-1}x^3+b_{n-1}x^2+c_{n-1}x+d_{n-1}) \Big|_{x=x_{n-1}} = \frac{d}{dx}(a_nx^3+b_nx^2+c_nx+d_n) \Big|_{x=x_{n-1}}$

$(3a_{n-1}x_{n-1}^2+2b_{n-1}x_{n-1}+c_{n-1}) = (3a_nx_{n-1}^2+2b_nx_{n-1}+c_n)$

$(3a_{n-1}- 3a_n)x_{n-1}^2+(2b_{n-1}- 2b_n)x_{n-1}+ (c_{n-1}- c_n)=0$

So we will get n-1 equations by equating the slopes at n-1 interior points.

Second derivative at two consecutive cubic are continuous at common interior points

$\frac{d^2}{dx^2}(a_1x^3+b_1x^2+c_1x+d_1) \Big|_{x=x_1} = \frac{d^2}{dx^2}(a_2x^3+b_2x^2+c_2x+d_2) \Big|_{x=x_1}$

$\frac{d}{dx}(3a_1x^2+2b_1x+c_1) \Big|_{x=x_1} = \frac{d}{dx}(3a_2x^2+2b_2x+c_2) \Big|_{x=x_1}$

$(6a_1x+2b_1) \Big|_{x=x_1} = (6a_2x+2b_2) \Big|_{x=x_1}$

$(6a_1x_1+2b_1) = (6a_2x_1+2b_2)$

$(6a_1-6a2)x_1+2b_1-2b_2=0$

$\frac{d^2}{dx^2}(a_{n-1}x^3+b_{n-1}x^2+c_{n-1}x+d_{n-1}) \Big|_{x=x_{n-1}} = \frac{d^2}{dx^2}(a_nx^3+b_nx^2+c_nx+d_n) \Big|_{x=x_{n-1}}$

$\frac{d}{dx}(3a_{n-1}x^2+2b_{n-1}x+c_{n-1}) \Big|_{x=x_{n-1}} = \frac{d}{dx}(3a_nx^2+2b_nx+c_n) \Big|_{x=x_{n-1}}$

$(6a_{n-1}x+2b_{n-1}) \Big|_{x=x_{n-1}} = (6a_nx+2b_n) \Big|_{x=x_{n-1}}$

$(6a_{n-1}x_{n-1}+2b_{n-1}) = (6a_nx_{n-1}+2b_n)$

$(6a_{n-1}-6a_n)x_{n-1}+2b_{n-1}-2b_n=0$

So we will get n-1 equations by equating the second derivative at n-1 interior points.

So far, **we have 2n+ (n-1)+(n-1) =4n-2 equation's** so we are left with 2 equations.

For last 2 equations we will use natural spline

Natural spline is defined as 2nd derivative for first and last polynomial equal to zero in the interpolation function boundary point .

$6a_1x_0 + 2b_1 = 0$

$6a_nx_n + 2b_n = 0$

So far we have 4n equations for n+1 data points

Numerical integral

The area bounded by curve f(x) and x-axis between limits a&b . It is denoted by

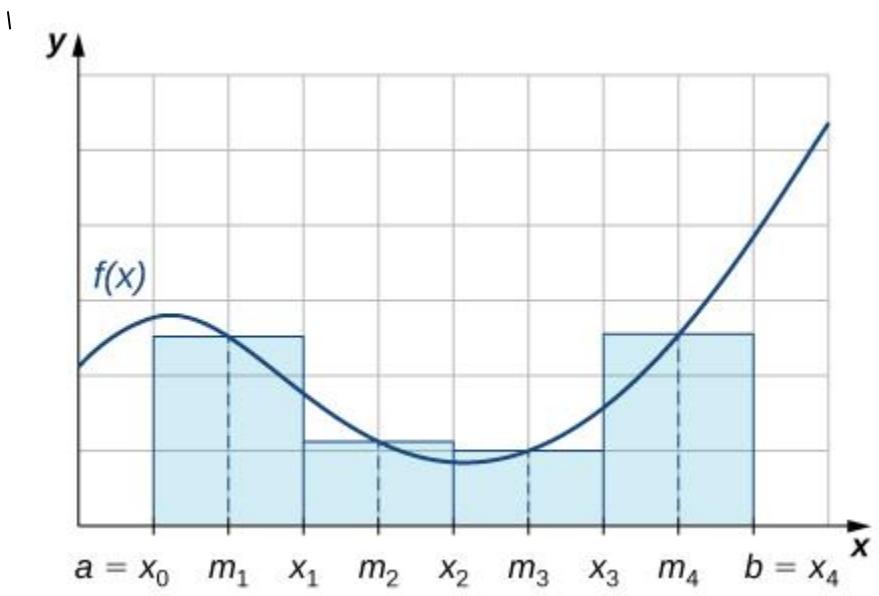
$I = \int_a^b f(x)dx$ .....eq 1

Divide interval (a,b) into n equal interval with length h (step size)$h = \frac{b-a}{n}$

$a = x_0$

$x_1 = x_0 +$

$.x_n = x_n + h$

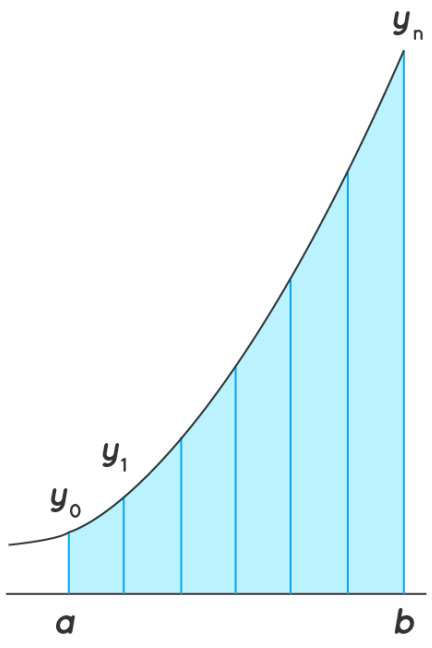


Equation 1 can be evaluated using

Trapezoid rule

$\int_a^b f(x)dx = h[(\frac{y_0+y_n}{2}) + y_1 + y_2 + \ldots\ldots\ldots y_n]$

Trapezoidal Rule Formula



$$\text{Area} = \int_a^b y\,dx \approx \frac{1}{2}h[y_0+2(y_1+y_2+\ldots+y_{n-1})+y_n]$$

$$\text{where } h = \frac{b-a}{n}$$

Question $\int_0^6 \frac{dx}{1+x^2}$

$h=\frac{b-a}{n}=\frac{6-0}{6}=1$

$x_0=0$(lower limit)

$x_1=x_0+h=0+1=1$

x	values
x_0	0
x_1	1
x_2	2
x_3	3
x_4	4
x_5	5
x_6	6

Now put all values in formula

$\int_0^6 \frac{dx}{1+x^2}=h[(\frac{y_0+y_n}{h})+y_1+y_2+\ldots\ldots\ldots y_n]\ldots\ldots\ldots\ldots$eq 1

X	0	1	2	3	4	5	6
y	1	0.5	0.2	0.1	0.05	0.038	0.027

$\int_0^6 \frac{dx}{1+x^2}$=1[($\frac{1+0.027}{2}$)+0.5+0.2+0.1+0.05+0.038+0.027=1.4285]

Error=$\frac{(b-a)^3}{12n^2}$|max f''(x) |x belong to (a,b)

F''(x)=|-$\frac{2x}{(x^2+1)^2}$|= $\frac{2*2}{25}$=0.16

Error will be

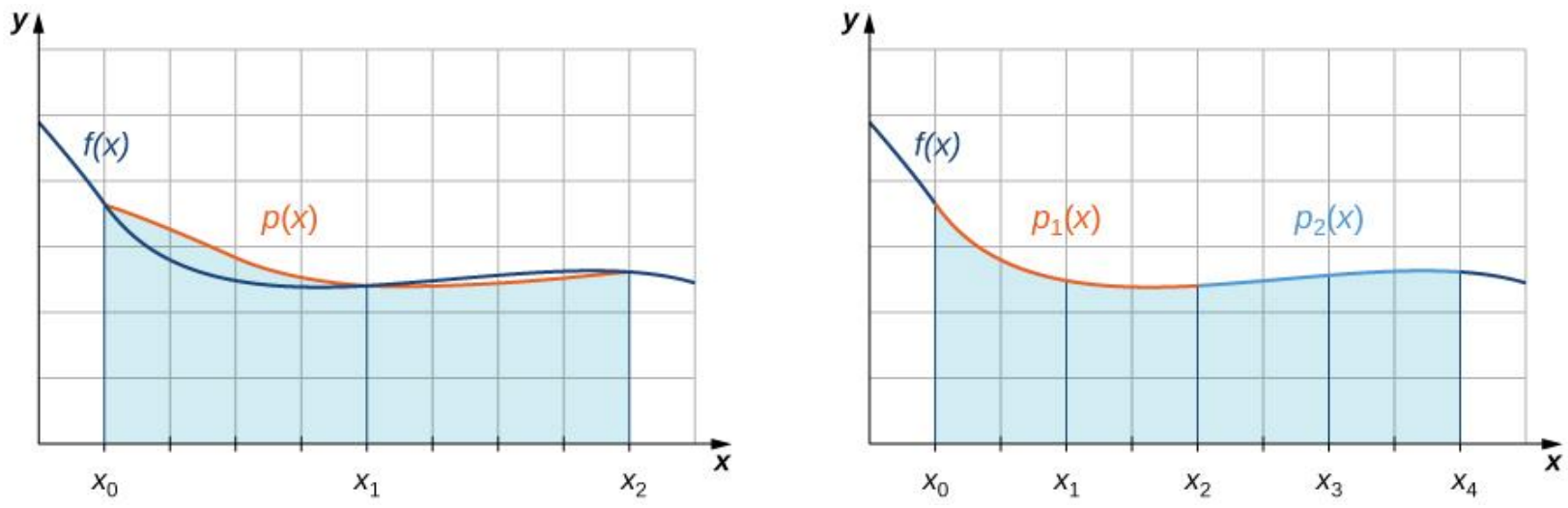
$\frac{6}{12(6)^2}$0.16=0.002

Generalized error

$\int_0^6 \frac{dx}{1+x^2} - I$

80.5-1.428579%

Simpson $\frac{1}{3^{rd}}$rule



It is applicable when total no of interval are even

$\int_a^b f(x)dx = \frac{h}{3}$(y0+yn+4(y1+y3...)+2(y2+y4+.....)

Question $\int_0^6 \frac{dx}{1+x^2}$

h=$\frac{b-a}{n}$=$\frac{6-0}{6}$=1

$\int_0^6 \frac{dx}{1+x^2}$=$\frac{1}{3}$ (1+0.027+4(0.5+0.1+0.038)+2(0.2+0.05+0.027)=1.377

Error=$\frac{(b-a)^3}{90}$|max $f^3(x)$ |x belong to (a,b)

max $f^3(x)$:x belong to (a,b)

$F^{3}(x) = -\frac{24x(x^2-1)}{(x^2+1)^4}$ =0.0576 atx=3

Error=$\frac{(b-a)^3}{90}$|max $f^3(x)$ |x belong to (a,b)

$\frac{6^3}{90}$*0.0576=0.1384

Generalized error

$|\int_0^6 \frac{dx}{1+x^2}$-SI|=80.5-1.3772=79.1

Simpson $\frac{3}{8}$rule

It is applicable if total no of interval is multiple of 3…

$\int_a^b \boldsymbol{f(x)dx} = \frac{3h}{8}[(y_0+y_n)+3(y_1+y_2+\ldots\ldots)+2(y_3+y_6+\ldots\ldots)$

$\int_0^6 \frac{dx}{1+x^2} = \frac{3*1}{8}[(1+0.027)+3(0.5+0.2+0.05+0.038)+2(0.1)=1.3571$

Error=$\frac{(b-a)^5}{180}$|max $f^4(x)$ |x belong to (a,b)

$\frac{6^5}{180}*\frac{24(5x^4-10x^2+1)}{(x^2+1)^5}$at x=3

=3.27

Generalized error=$|\int_0^6 \frac{dx}{1+x^2}$-SI|=|80.95-1.3571|=7

Analyze and understand that in a practical situation methods must not only be justified by its order of convergence. Other characteristics can become important as the number of iterations, therefore the comparison of two or more methods should take into account the order of

convergence, computational cost (CPU time used) asymptotic constant of the error and the value of the initial approach used for the method.

Numerical methods allow us to find approximate results to complex systems by applying simple mathematical operations.

Open methods converge faster than closed methods, by using an approximate value closest to the root, without starting at an interval